

# Urządzenia Peryferyjne Laboratorium

08.11.2023

Grupa nr 7B, Czwartek, godz. 11:00, tydzień nieparzysty

Autorzy:

264488 Adam Czekalski

252560 Karol Szydłak

nr	Treść zadania	data wykonania
2.	Ćwiczenie 5 Joystick. Sterowanie kursorem.	26/10/2023

# 1. Wstęp teoretyczny

W trakcie laboratorium zrobiono aplikację z interfejsem graficznym pozwalającą na:

- wyliczanie urządzeń podpiętych do komputera i aktywację wybranych
- wyliczanie elementów/osi joysticka i przyporządkowywanie im zakresów
- odczytywanie stanu (przyciski, potencjometry, pozycja drążka) joysticka
- przejęcie działania myszy, sterowanie kursorem oraz rysowanie na ekranie za pomocą joysticka

Ogólnie rzecz biorąc rozróżniamy trzy koncepcje:

- HAL – (ang. Hardware Abstraction Layer) zestaw rutyn w oprogramowaniu, które zapewniają programom dostęp do sprzętowych zasobów poprzez programowalne interfejsy
- HEL – (ang. Hardware Emulation Layer) warstwa emulacji sprzętu pozwala na uruchomienie oprogramowania napisanego na daną platformę na innej architekturze sprzętowej.
- HID – (ang. Human Interface Device) termin używany do opisu urządzeń, umożliwiających komunikację człowieka z komputerem. Urządzenia te przekazują dane od użytkownika do urządzenia docelowego.

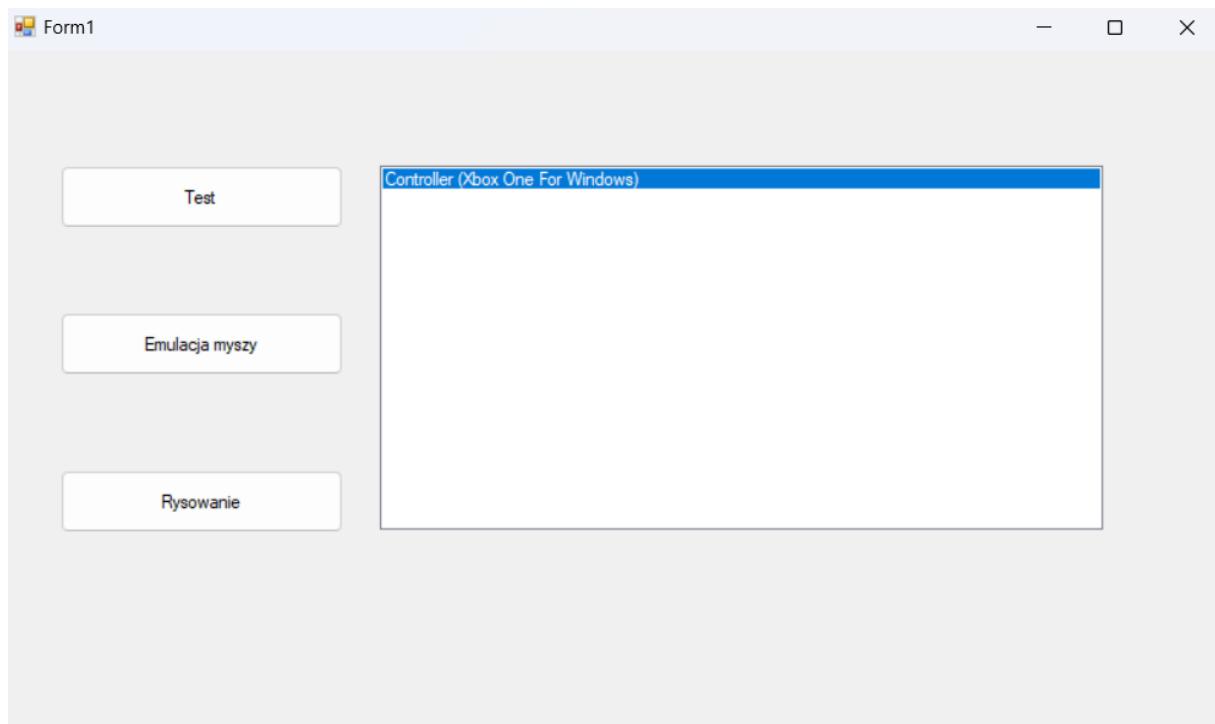
Aby móc sterować joystickiem, można wykorzystać bibliotekę DirectInput.

Budowa i zasada działania kontrolerów

- Mysz:  
Najstarszym rozwiązaniem jest mysz mechaniczna. Zawiera ona metalową kulę oraz system rolek. Tarcie kuli o podłoże obraca rolki, co jest następnie interpretowane elektronicznie i transformowane na ruch kursorem. Nowszym rozwiązaniem jest mysz optyczna, w którą wbudowany jest czujnik optyczny rejestrujący ruch myszą. Urządzenie zawiera także kilka przycisków reprezentujących różne akcje, takie jak: kliknięcie lewym lub prawym przyciskiem myszy.
- Joystick  
Urządzenie składa się z (zazwyczaj analogowego) drążka wychylnego oraz zestawu przycisków. Drążek joysticka cyfrowego pozwala jedynie na wybór kierunku ruchu z zestawu dostępnych, natomiast drążek joysticka analogowego na określenie kierunku ruchu z zakresu 360°, a także na określenie jego „intensywności”. Drążek zazwyczaj posiada dwie niezależne osie ruchu X oraz Y. Pozycja drążka na osi jest reprezentowana przez 16 bitową liczbę, gdzie wartość środkowa 32767 jest pozycją neutralną na osi i reprezentuje brak wychylenia drążka.
- Gamepad  
Kontroler posiadający wiele przycisków i od jednego do kilku joysticków.

# 2. Aplikacja

Aplikacja do obsługi joysticka została napisana w języku C#, korzystając z biblioteki DirectInput oraz z Windows Forms w celu stworzenia interfejsu graficznego



Rysunek 2-1 Menu główne aplikacji

## 2.1. Wyliczenie urządzeń podpiętych do komputera i aktywacja wybranych

```
SharpDX.DirectInput.DirectInput directInput = new SharpDX.DirectInput.DirectInput();
List<DeviceInstance> deviceList = new List<DeviceInstance>(); //stworzenie listy urządzeń
1 reference
public Menu()
{
    InitializeComponent();

    var joysticks = directInput.GetDevices(DeviceType.Joystick, DeviceEnumerationFlags.AttachedOnly);
    var gamepads = directInput.GetDevices(DeviceType.Gamepad, DeviceEnumerationFlags.AttachedOnly);
    //poniżej wyświetlanie podłączonych urządzeń
    foreach(DeviceInstance instance in joysticks)
    {
        deviceList.Add(instance);
        listBox1.Items.Add(instance.InstanceName);
    }

    foreach (DeviceInstance instance in gamepads)
    {
        deviceList.Add(instance);
        listBox1.Items.Add(instance.InstanceName);
    }
}
```

Rysunek 2. Kod odpowiedzialny za wyświetlenie listy urządzeń podpiętych do komputera.

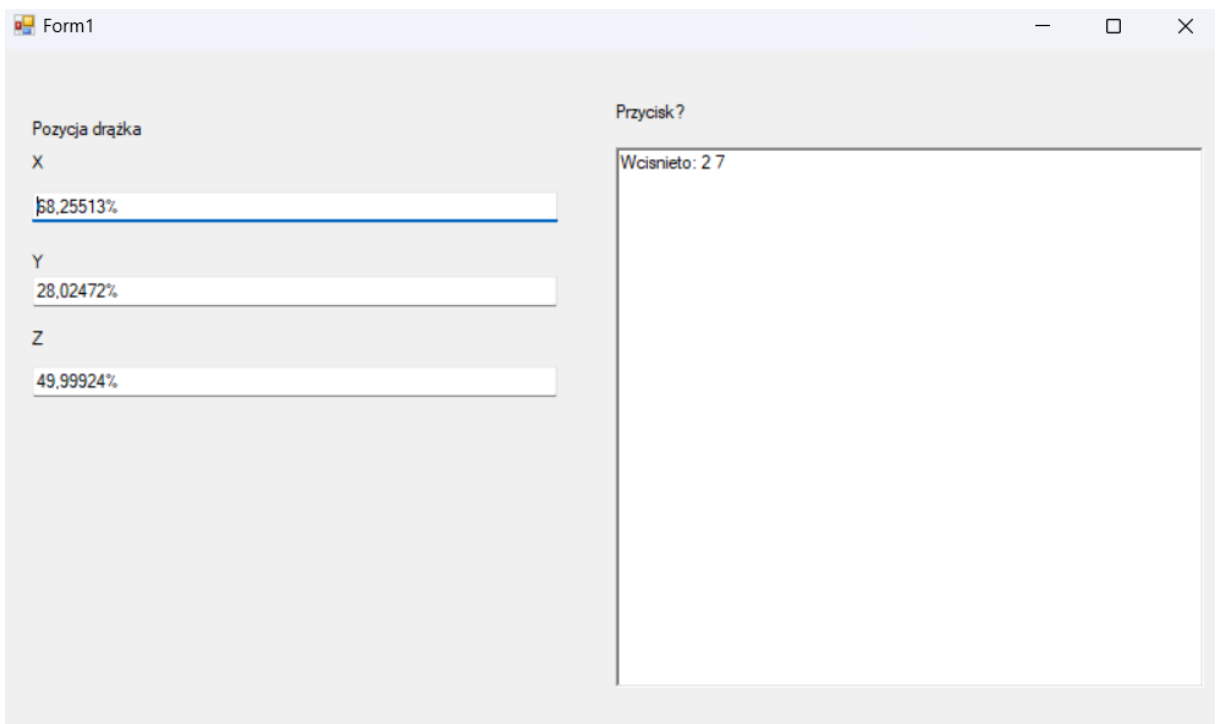
W konstruktorze pobieramy listę podłączonych do komputera joysticków oraz gamepadów. Następnie iterujemy przez obie Listy i dodajemy urządzenia do jednej Listy oraz listBoxa, aby w aplikacji móc wybrać żądane urządzenie.

## 2.2. Odczytywanie stanu (przyciski, potencjometry, pozycja drążka) joysticka

```
1 reference
private void timer1_Tick(object sender, EventArgs e)
{
    //poniżej wyświetlenie informacji o wciśniętych przyciskach
    richTextBox1.Clear();
    richTextBox1.Text += "Wcisnieto: ";
    for (int i = 0; i < joystick.GetCurrentState().Buttons.Count(); i++)
    {
        if (joystick.GetCurrentState().Buttons[i]) {
            richTextBox1.Text += i.ToString() + " ";
        }
    }
    //poniżej odczytanie pozycji drążka
    int inputOffset = 32767;
    textBox1.Text = String.Format("{0}%", (float)joystick.GetCurrentState().X / inputOffset);
    textBox3.Text = String.Format("{0}%", (float)joystick.GetCurrentState().Y / inputOffset);
    textBox4.Text = String.Format("{0}%", (float)joystick.GetCurrentState().Z / inputOffset);
}
```

Rysunek 3. Kod odpowiedzialny za odczyt stanu joysticka

Okienko WindowsForms posiada timer, który odświeża się co 10ms. Za każdym odświeżeniem, wywołuje powyższą funkcję. Pętla wewnątrz funkcji wykonuje się tyle razy, ile joystick posiada przycisków. Aby uzyskać ich liczbę, należy użyć metody *GetCurrentState()*. Następnie jeśli dany przycisk jest wciśnięty, to wypisujemy jego numer w okienku w GUI. Następnie odczytujemy pozycję drążka. Tu także korzystamy z metody *GetCurrentState()*, aby uzyskać zakres X i Y drążka. Następnie uzyskaną wartość dzielimy przez wartość środkową, aby uzyskać wartość procentową wychylenia drążka. Podobnie z osią Z drążka, która w przypadku joysticka na laboratorium odczytywała pozycję suwaka.



Rysunek 4. Test joysticka

### 3. Emulacja myszy

Emulacja myszy odbywa się poprzez odczytanie wartości dla osi X oraz Y joysticka, a następnie obliczenie przesunięcia myszy i aktualizowanie na ich podstawie przesunięcia kursora myszy. Naciśnięcie odpowiednich przycisków na joysticku („0” - lewy przycisk myszy, „1” - prawy przycisk myszy) powoduje ustawienie odpowiedniej flagi zdarzenia myszy i wywołanie metody „mouse\_event” odpowiedzialnej za symulację działania myszy, funkcja ta została zaimportowana z biblioteki „user32.dll”.

Poniższy rysunek przedstawia część metody odpowiedzialną za aktualizację położenia kursora myszy

```
24 public void InitializeEmulation()
25 {
26     uint flags = (uint)(MouseEventFlags.ABSOLUTE | MouseEventFlags.MOVE); // flagi umożliwiające śledzenie ruchu myszy(MOVE) oraz
27     //ustawienie flagi Absolute powoduje że współrzędne myszy przekazywane w zdarzeniach są wartościami bezwzględными
28     //mysz jest śledzona w zakresie całego ekranu, nie tylko aplikacji
29     int inputOffset = 32767; // input offset = 32767 (pozycja joysticka jest reprezentowana przez 16 bitową liczbę
30     //gdzie wartość 32767 jest wartością środkową, musi zostać odjęta od odczytanej wartości wychylenia drążka, aby w pozycji neutralnej dać 0)
31     while (true)
32     {
33
34         int x = joystick.GetCurrentState().X - inputOffset;
35         int y = joystick.GetCurrentState().Y - inputOffset;
36         //wyliczenie przesunięcia na osiach x i y
37         float xOffset = (float)x / inputOffset;
38         float yOffset = (float)y / inputOffset;
39
40         if ((xOffset < 0.25 && yOffset < 0.25)&&(xOffset > -0.25 && yOffset > -0.25)) //ponieważ drążki są analogowe w pozycji neutralnej
41             //odczytana wartość jest bliska 0, ale niekoniecznie równa stąd zaokrąglenie aby podczas emulacji kursor nie poruszył się
42             //gdy drążek jest w pozycji neutralnej
43         {
44             xOffset = 0;
45             yOffset = 0;
46         }
47         else
48         {
49             xOffset *= 2;
50             yOffset *= 2;
51         }
52
53         mouseX += xOffset;
54         mouseY += yOffset;
55     }
56 }
```

Rysunek 5. Aktualizacja położenia kursora myszy

Wewnątrz pętli obliczane jest położenie kursora na obu osiach pomniejszone o inputOffset. Zmienna ta reprezentuje wartość odczytywaną, gdy drążek analogowy nie jest wychylony, wobec czego ta korekta jest konieczna aby emulacja działała poprawnie. Następnie obliczana jest względna zmiana położenia na osiach względem stanu poprzedniego (xOffset, yOffset). Ponieważ drążki joysticka jest analogowe, po obliczeniu wychylenia drążka gdy jest on w stanie neutralnym otrzymamy wartości bliskie 0. Dlatego wartości w przedziale [-0.25,0.25] są nadpisywane 0, w przeciwnym wypadku ruch myszy zaobserwowany byłby przy pozycji neutralnej drążka.

Poniżej przedstawiono dalszą część metody odpowiedzialną za obsługę lewego przycisku myszy

```
56
57
58     if (joystick.GetCurrentState().Buttons[0])
59     {
60         flags |= (uint)MouseEventFlags.LEFTDOWN; //lewy przycisk myszy wciśnięty
61
62         if (joystickInUse)
63             mouse_event(flags, (uint)mouseX, (uint)mouseY, 0, 0);
64     }
65     else
66     {
67         flags |= (uint)MouseEventFlags.LEFTUP; //lewy przycisk myszy zwolniony
68         if (joystickInUse)
69             mouse_event(flags, (uint)mouseX, (uint)mouseY, 0, 0);
70
71         flags = (uint)(MouseEventFlags.ABSOLUTE | MouseEventFlags.MOVE);
72         if (joystickInUse)
73             mouse_event(flags, (uint)mouseX, (uint)mouseY, 0, 0);
74     }
75
76     if (joystick.GetCurrentState().Buttons[1])
77     {
78         flags |= (uint)MouseEventFlags.RIGHTDOWN;
79         if (joystickInUse)
80             mouse_event(flags, (uint)mouseX, (uint)mouseY, 0, 0);
81     }
82     else
83     {
84         flags |= (uint)MouseEventFlags.RIGHTUP;
85         if (joystickInUse)
86             mouse_event(flags, (uint)mouseX, (uint)mouseY, 0, 0);
87     }
```

Rysunek 6. Emulacja przycisków myszy

Wciśnięcie przycisku „0” na joysticku oraz jego zwolnienie powoduje ustawienie odpowiednich flag, a następnie wywołanie metody „mouse\_event” odpowiedzialnej za obsługę zdarzenia myszy, co emuluje naciśnięcie oraz zwolnienie lewego przycisku myszy. Analogicznie postępuje się dla prawego przycisku myszy, obsługiwanego przez przycisk „1”.

Aby rozpocząć emulację myszy należy wcisnąć przycisk „2”, aby ją zakończyć przycisk „3”. Wraz z definicją flag zdarzeń myszy zostało to pokazane na poniższym rysunku.

```
89  if (joystick.GetCurrentState().Buttons[2])// przycisk uruchamia emulację myszy
90  {
91      joystickInUse = !joystickInUse;
92  }
93
94
95  if (joystick.GetCurrentState().Buttons[3])
96  {
97      break;
98  }
99
100
101  if (joystickInUse)
102      mouse_event(flags, (uint)mouseX, (uint)mouseY, 0, 0);
103
104  }
105
106  }
107
108  [Flags] // definicja flag
109  Odwołania: 8
110  public enum MouseEventFlags : uint
111  {
112      LEFTDOWN = 0x00000002,
113      LEFTUP = 0x00000004,
114      MOVE = 0x00000001,
115      ABSOLUTE = 0x00000000,
116      RIGHTDOWN = 0x00000008,
117      RIGHTUP = 0x00000010
118  }
119
120  [DllImport("user32.dll")] //dll używany do importu funkcji mouse_event używanej do obsługi zdarzenia myszy
121
122  Odwołania: 6
123  static extern void mouse_event(uint dwFlags, uint dx, uint dy, uint dwData, int dwExtraInfo);
124 }
```

Rysunek 7. Flagi zdarzeń myszy

## 4. Rysowanie na ekranie

Aplikacja umożliwi uruchomienie okna, pozwalającego na rysowanie na ekranie za pomocą myszy lub joysticka. Kod odpowiedzialny za to okno przedstawiają poniższe rysunki.

```
13 public partial class Paint : Form // okno pozwalające na rysowanie
14 {
15     Graphics g;
16     int x = -1;
17     int y = -1;
18     bool moving = false;
19     Pen pen;
20     1 odwołanie
21     public Paint()
22     {
23         InitializeComponent();
24         g = panel1.CreateGraphics();
25         g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
26         pen = new Pen(Color.Black, 5);
27         pen.StartCap = pen.EndCap = System.Drawing.Drawing2D.LineCap.Round;
28     }
29     1 odwołanie
30     private void panel1_Paint(object sender, PaintEventArgs e)
31     {
32         //PictureBox p = (PictureBox)sender;
33     }
34     1 odwołanie
35     private void panel1_MouseDown(object sender, MouseEventArgs e)
36     {
37         moving = true;
38         x = e.X;
39         y = e.Y;
40         panel1.Cursor = Cursors.Cross;
41     }
```

Rysunek 8. Kod odpowiedzialny za rysowanie na ekranie

```
41
42 1 odwołanie
43 private void panel1_MouseMove(object sender, MouseEventArgs e)
44 {
45     if(moving && x!=-1 && y!= -1)
46     {
47         g.DrawLine(pen, new Point(x, y), e.Location);
48         x = e.X;
49         y = e.Y;
50     }
51
52     1 odwołanie
53 private void panel1_MouseUp(object sender, MouseEventArgs e)
54 {
55     moving = false;
56     x = -1;
57     y = -1;
58     panel1.Cursor = Cursors.Default;
59 }
60 }
```

Rysunek 9. Kod odpowiedzialny za rysowanie na ekranie



Powyższy kod tworzy panel, w którym gdy trzymamy lewy przycisk myszy rysujemy linie dzięki metodzie `panel1_MouseDown`, poruszanie myszką bez wciśnięcia lewego przycisku myszy powoduje jedynie ruch kursora dzięki metodzie `panel1_MouseUp`.

Poniższy rysunek przedstawia uruchomione okno rysowania.



Rysunek 10. Okno pozwalające na rysowanie

## 5. Źródła

- 1) [https://pl.wikipedia.org/wiki/Mysz\\_komputerowa](https://pl.wikipedia.org/wiki/Mysz_komputerowa)
- 2) <https://pl.wikipedia.org/wiki/Gamepad>
- 3) <https://pl.wikipedia.org/wiki/D%C5%BCojstik>